Improving Weight-Inherited Distillation with Data-aware Initialization and Structural Adaptation

Anonymous ACL submission

Abstract

Weight-Inherited Distillation (WID) is an effective distillation method that inherits the weights from the teacher's model, thus achieving better results than traditional distillation methods. However, the identity matrix initialization used in WID leads to slow model convergence. In this work, we propose an improved WID method named DA-WID that replaces the identity matrix initialization with a specialized data-aware initialization. Concurrently, we refine the structural design of WID, enhancing its adaptability and flexibility in selecting the compressed model architecture. Our experiments on the GLUE and SQuAD datasets show that the model delivered by DA-WID retains 96% of the performance with 94% of parameters removed, showing its effectiveness compared to previous pruning and distillation methods. Our data and code is available on an anonymous repo.

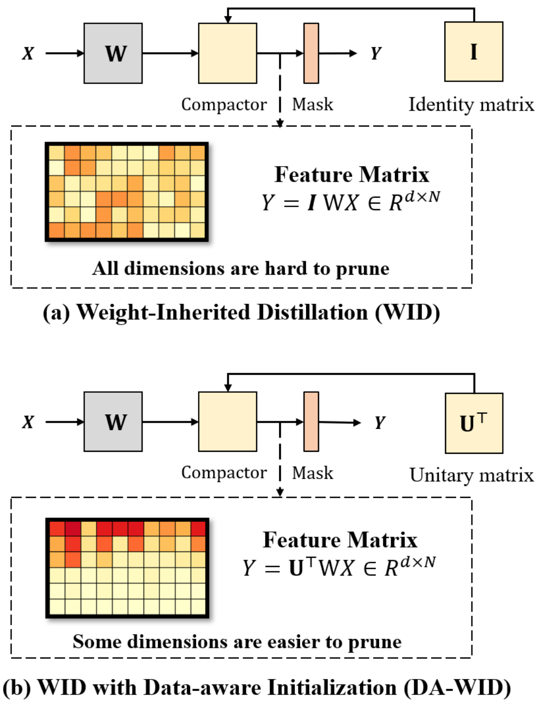


Figure 1: Comparing the fundamental concepts of WID and DA-WID: (a) In WID, compactor matrix and mask are used to compress the output dimension of the weight matrix W. However, initializing with the identity matrix poses challenges for feature filtering using the mask. (b) In DA-WID, initializing the compactor matrix using the unitary matrix derived from SVD simplifies feature filtering with the mask.

1 Introduction

Pre-trained language models (PLMs) (Devlin et al., 2018; Radford et al.; Vaswani et al., 2017) have gained widespread use in Natural Language Processing due to their remarkable performance. However, their considerable storage needs and extended inference durations can complicate real-world deployments. To address these limitations, significant efforts have been made to make PLMs both smaller and faster. Among these methods, distillation has emerged as a popular approach.

Distillation offers a versatile technique for model compression. It permits a custom specification of the student model's structure and facilitates the gradual transfer of knowledge from the larger teacher model during training. A notable drawback, however, is that distillation often doesn't leverage the full potential of the teacher model's parameters. As a result, student models typically require pre-training on a vast, unlabeled dataset before being fine-tuned for specific tasks. A novel distillation method called weight-inherited distillation (WID) (Wu et al., 2023) has been introduced recently. WID incorporates compactor matrices into the teacher model and achieves compression through re-parameterization. Distinct from traditional distillation techniques, WID allows for a direct inheritance of the teacher model's parameters, eliminating the need to train the student model from scratch.

Nonetheless, the WID methodology has its inherent limitations. The approach employs identity matrices for the initiation of compactor matrices, which presents challenges in optimization when seeking the desired compression. As shown in Fig.1 (a), the input feature X is successively passed through the weight matrix and the compactor matrix, while the mask is responsible for pruning the useless dimensions from the features. When the compactor matrix is initialized with the identity matrix, the activation values are dispersed across all dimensions of the feature matrix, increasing the difficulty of eliminating irrelevant dimensions.

To address WID's challenges, we introduce DA-WID, a distinct distillation method emphasizing Data-Aware initialization for compactor matrices. We postulate that the compactor matrix in WID should primarily extract the primary components of the features. As shown in Fig.1 (b), we expect the initialization centers the activation values predominantly on select dimensions of the feature matrix, considerably simplifying the task of filtering out irrelevant dimensions. With this in mind, we initialize the compactor matrices to inherently possess this extraction capability even before the training process. Specifically, we innovatively draws a connection between the WID methodology and the low-rank properties of features, setting unique low-rank approximation objectives for different PLM modules. We then initialize the compactor matrices based on these approximation outcomes.

Moreover, existing pruning work (Xia et al., 2022) shows that the structure of various layers of the model tends to be different after pruning. Thus, we refine WID's model structure to allow dynamic determination of the student model's structure, guided by the desired sparsity during optimization. At the same time, we also integrate WID with pruning to improve the flexibility of WID.

We conducted comprehensive experiments on the GLUE and SQuAD benchmarks. Our findings highlight three primary advantages: (1) Our approach compressed 94% of the parameters in the BERT-base model, decreasing its size from 85M to 5M, yet retained an accuracy of over 96%. (2) Compared to baseline methods, our technique produces models with superior accuracy at comparable compression rates. (3) Ablation studies indicate that both data-aware initialization and structural adaptation notably enhance the accuracy of the compressed model.

The contributions can be summarized as:

- We propose a data-aware initialization method that reduces the difficulty of optimizing the compactor matrix in WID.
- We improve the WID structure so that WID can adaptively set the structure of the compression model according to the desired sparsity.

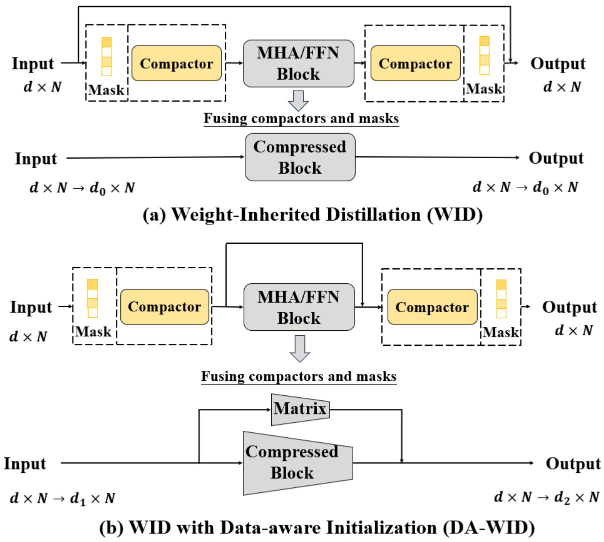


Figure 2: Illustration of the basic structure of WID (Wu et al., 2023) and DA-WID. (a) WID places the compactor matrices and masks inside the residuals, resulting in the input and output dimensions being compressed to the same dimension. (b) DA-WID places the compactor matrices and masks outside the residuals, thus allowing the dimensions of the input and output to be compressed into different dimensions.

2 Background

2.1 Pre-trained Language Model

A typical PLM (e.g., BERT (Devlin et al., 2018)) comprises a stack of transformer layers, each containing a multi-head attention (MHA) block and a feed-forward network (FFN) block.

Multi Head Attention (MHA). The MHA block takes $x \in \mathbb{R}^{d \times N}$ as input and consists of H attention heads that facilitate interactions between tokens, with a normalization (Ba et al., 2016) step.

$$x_{\mathrm{M}} = \mathrm{LN}(x + \mathrm{MHA}(x)), \tag{1}$$

$$\mathrm{MHA}(x) = \sum_{i=1}^{H} \mathrm{Att}^{(i)}(x), \tag{2}$$

$$\mathrm{Att}^{(i)}(x) = W_O^{(i)\top} W_V^{(i)} x \cdot \mathrm{Softmax}((W_K^{(i)} x)^\top (W_Q^{(i)} x)/\sqrt{d_h}), \tag{3}$$

where $W_Q^{(i)}, W_K^{(i)} \in \mathbb{R}^{d_{\mathrm{QK}} \times d}$, $W_V^{(i)}, W_O^{(i)} \in \mathbb{R}^{d_{\mathrm{VO}} \times d}$ are the parameters of an MHA block, denoting the query, key, value, and output matrices,

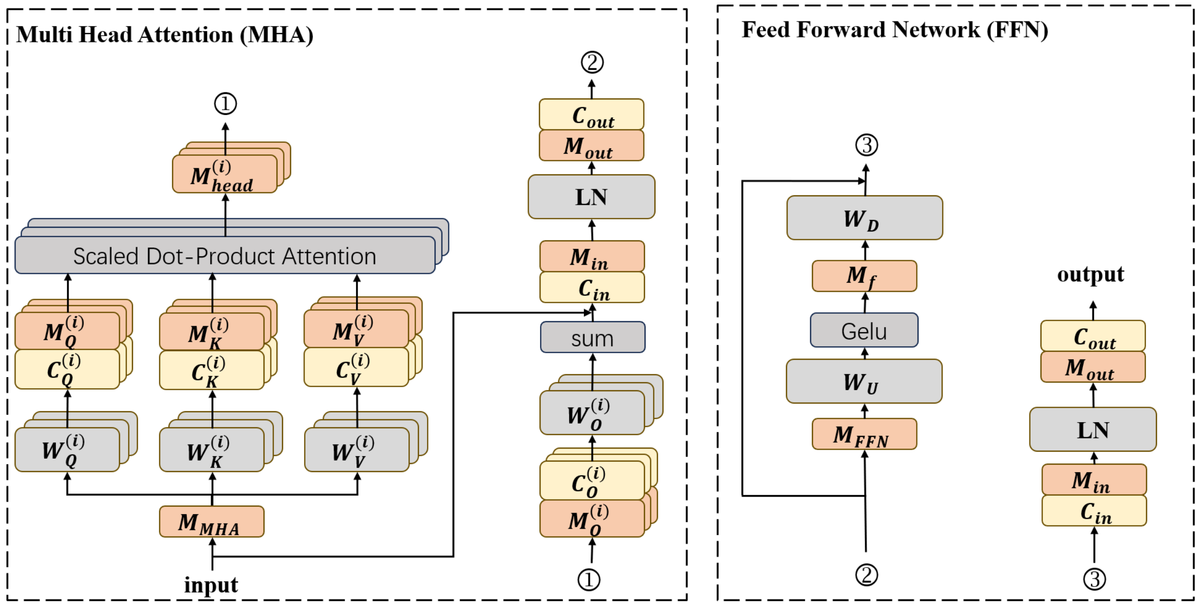


Figure 3: Illustration of the DA-WID structure, where the gray rectangles denote the weight matrices, the yellow rectangles denote the compactor matrices, and the red rectangles denote the masks.

respectively. Here d denotes the hidden dimension, $d_h = d/H$ denotes the head size, N denotes the sequence length, and d_{QK} and d_{VO} denote the intermediate dimensions of the MHA block.

Feed Forward Network (FFN). The FFN block takes x_M as input and generates x_F as output.

$$x_F = \text{LN}(x_M + \text{FFN}(x_M)), \tag{4}$$

$$\text{FFN}(x_M) = W_D^\top \text{gelu}(W_U x_M), \tag{5}$$

where gelu is the activation function, and $W_U, W_D \in \mathbb{R}^{d_f \times d}$ are two weight matrices of the FFN block. Here, d_f indicates the intermediate dimension of the FFN block.

2.2 Weight-Inherited Distillation

WID is a distillation method that, in contrast to conventional distillation methods, inherits the teacher model's parameters and compresses them.

Figure 2 (a) illustrates the core concept of WID's model compression. Initially, WID introduces compactor matrices $W_L \in \mathbb{R}^{d\times d}$ and $W_R \in \mathbb{R}^{d\times d}$, along with associated masks $M_L \in \mathbb{R}^d$ and $M_R \in \mathbb{R}^d$, positioned on either side of the transformer block. The compactor matrices are set to the identity matrix at the start, while the masks begin as vectors filled with ones. After this setup, the model undergoes training using a large dataset, and compactor matrices are progressively pruned based on the masks. The final step involves fusing the transformer block with the compactors to produce a compressed transformer block. It can be seen that the input and output of the transformer block are compressed from d to d_0 dimension.

3 Method

In this section, we begin by discussing improvements made to the WID structure, enabling it to selectively choose the compressed model structure (Section 3.1). We then delve into the data-aware initialization, which can benefit model optimization compared to identity matrix initialization. Our focus is on explaining how to enhance the compactor matrix's capability to extract primary feature components prior to training (Section 3.2). We conclude by detailing the training (Section 3.3) and fusing (Section 3.4) procedures of DA-WID .

3.1 Structure of DA-WID

The structure of DA-WID is shown in Fig. 3. On top of the MHA block, we insert the compactor matrices C_Q, C_K, C_V, and C_O and the corresponding masks M_Q, M_K, M_V, and M_O for compressing the intermediate dimensions d_{QK}, d_{VO} of the MHA blocks. At the position of LayerNorm layers, we insert the compactor matrices $C_{\text{in}}, C_{\text{out}}$ and the corresponding masks $M_{\text{in}}, M_{\text{out}}$ for compressing the hidden dimensions d between layers. At the FFN block, we introduce the mask M_f for compressing the intermediate dimension d_f of the FFN block. We don't insert the compactor matrices in the FFN block because existing pruning work has shown that the intermediate dimensions of the FFN block can be effectively compressed by using only masks. Since the above masks are used to compress the input and output dimensions of the weight matrix, we refer to the above masks as **dimension-level** masks.

We also introduce structure pruning to further enhance the flexibility of the WID method. At the MHA block, we introduce a **head-level** mask M_{head} to reduce the attention head size. Finally, we introduce **layer-level** masks $M_{\text{MHA}}, M_{\text{FFN}}$ for removing the entire MHA block or FFN block.

Compared to WID, we change the position of the compactor matrix with respect to the residuals, which allows our method to adaptively select the hidden dimension of different layers. As shown in Fig. 2, since WID places the compactor matrices and masks inside the residuals, the input and the output dimensions of the compressed MHA/FFN block are required to be the same. On the contrary, since DA-WID places the compactor matrices and masks outside the residuals, the input and output dimensions of the compressed MHA/FFN block could be different. In addition, we also introduce the structured pruning into DA-WID , which extends the compression granularity and thus increases the flexibility of the WID method.

The computation process of DA-WID can be expressed as follows:

Multi Head Attention (MHA).

$$\begin{aligned}
&x_{\text{M}} = C_{\text{out}} M_{\text{out}} \text{LN}(M_{\text{in}} C_{\text{in}}(x + \text{MHA}(x))), \\
&\text{MHA}(x) = \sum_{i=1}^{H} M_{\text{head}}^{(i)} \text{Att}^{(i)}(x), \\
&\text{Att}^{(i)}(x) = W_O^{(i)\top} C_O^{(i)\top} M_O^{(i)\top} M_V^{(i)} C_V^{(i)} W_V^{(i)} x \cdot \\
&\text{Softmax}((W_K^{(i)} x)^\top C_K^{(i)\top} M_K^{(i)\top} M_Q^{(i)} C_Q^{(i)} (W_Q^{(i)} x)).
\end{aligned} \tag{6}$$

Feed Forward Network (FFN).

$$\begin{aligned}
&x_F = C_{\text{out}} M_{\text{out}} \text{LN}(M_{\text{in}} C_{\text{in}}(x_M + \text{FFN}(x_M))), \\
&\text{FFN}(x_M) = W_D^\top M_f \text{gelu}(W_U x_M),
\end{aligned} \tag{7}$$

where C_* represents the compactor matrix and M_* represents the vectors corresponding to the mask. When multiplying a mask vector with a matrix, we transfer the mask vector into a diagonal matrix. Note that the matrices $C_{\text{in}}, C_{\text{out}}$ and masks $M_{\text{in}}, M_{\text{out}}$ in the MHA block and the FFN block use different weights, but for the sake of simplicity, we do not distinguish between them in the formula.

3.2 Data-aware Compactor Initialization

WID initializes the compactor matrices to the identity matrices. It makes these compactor matrices hard to optimize for compression. Thus, our target is to find a better way for compactor initialization. We postulate that the compactor matrices in WID should extract the primary components of the features. Based on this conjecture, we present the initialization of compactors in the MHA block and the FFN layer, which encompasses compression of the intermediate dimensions d_{QK}, d_{VO} in the MHA block and the hidden layer dimension d in both the blocks.

Compressing d_{QK}, d_{VO}. The intermediate dimensions d_{QK}, d_{VO} are situated within the attention computation (i.e., Eq. 3) of the MHA block. Consequently, We formulate the following optimization objective functions to approximate the dot product result and the weighted average result of the attention component using their respective first k principal components:

$$\arg\min_{U_{k,0}, V_{k,0}} \|XW_Q^{(i)\top} W_K^{(i)} X - XW_Q^{(i)\top} V_{k,0}^{(i)\top} U_{k,0}^{(i)} W_K^{(i)} X\|, \tag{8}$$

$$\arg\min_{U_{k,1}, V_{k,1}} \|W_O^{(i)\top} W_V^{(i)} XS - W_O^{(i)\top} V_{k,1}^{(i)\top} U_{k,1}^{(i)} W_V^{(i)} XS\|, \tag{9}$$

where the feature matrix $X \in R^{d \times N}$ is derived from the last transformer layer for a specific calibration dataset. In the MHA block, S represents the Softmax of dot-product results. Here, N is the count of sampled tokens in the calibration dataset, while d signifies the feature dimension. Using SVD, we determine all the feature's principal components, resulting in matrices $U^{(i)}, V^{(i)}$. Here, matrices $U_{k,}^{(i)}, V_{k,}^{(i)} \in R^{k \times d}$ relate to the initial k columns of $U^{(i)}, V^{(i)}$. For a detailed breakdown of the SVD decomposition, see Appendix A. Note that while the optimization objective includes the number of principal components k, employing SVD to address this doesn't necessitate a predefined k—we are primarily interested in $U^{(i)}, V^{(i)}$.

Since $U_*^{(i)}, V_*^{(i)}$ have the property of distinguishing between the major and minor components of the features, allowing some dimensions to be pruned more easily, we consider these matrices to be a better choice than identity matrices. Specifically, we assign $C_Q^{(i)} = V_0^{(i)}, C_K^{(i)} = U_0^{(i)}$ and $C_O^{(i)} = V_1^{(i)}, C_V^{(i)} = U_1^{(i)}$.

Compressing d. For compressing the hidden dimension d between layers, We formulate the following optimization objective:

$$\arg\min_{U_k} \|X - U_k U_k^\top X\|, \tag{10}$$

where the feature matrix $X \in R^{d\times N}$ originates from either the attention computation (i.e., Eq. 2) with residual of the MHA block or the FFN computation (i.e., Eq. 5) with residual in the FFN block. It also serves as the input for the $\mathrm{LayerNorm}$ layer. The matrix U_k represents the first k columns of matrix U, which is derived from the SVD of matrix X, given by $U, \Sigma, V^\top = \mathrm{SVD}(X)$.

With this optimization objective, we can have

$$\begin{aligned}\mathrm{LN}(X) &= \gamma \hat{\mathrm{LN}}(X) + \beta \\ &= \gamma \hat{\mathrm{LN}}(UU^T X) + \beta \\ &\approx \gamma \hat{\mathrm{LN}}(U_k U_k^\top X) + \beta,\end{aligned} \tag{11}$$

where LN represents $\mathrm{LayerNorm}$, while $\hat{\mathrm{LN}}$ signifies $\mathrm{LayerNorm}$ without any weight or bias. Ideally, we aim to swap the order of computation between the matrices U and $\hat{\mathrm{LN}}$ for more effective fusion with subsequent weight matrices. Yet, this interchange would yield different outcomes, specifically, $\hat{\mathrm{LN}}(UU^T X) \neq U\hat{\mathrm{LN}}(U^\top X)$. Nevertheless, experimental observations reveal that in most layers, the discrepancies arising from this reordered computation can be rectified during model training. Thus, we propose the following approximation:

$$\mathrm{LN}(X) \approx \gamma U \hat{\mathrm{LN}}(U^\top X) + \beta. \tag{12}$$

It's note that just like $U^{(i)}$ and $V^{(i)}$, the inclusion of k in the objective function is merely to illustrate the low-rank property and doesn't necessitate specifying a value for k during optimization. In essence, we derive the matrix U using SVD and use it to initialize the compactor matrix on both sides of the $\mathrm{LayerNorm}$, including its weight and bias. To be specific, we assign $C_{\text{in}} = U^\top$ and $C_{\text{out}} = \gamma U$, accompanied by a bias of β.

3.3 Model Training

Pruning Objective. Model compression is controlled by a series of masks that are inserted into the model. We compute the sparsity of the model based on these masks. During training, all masking variables are learned as real numbers. At the end of training, masked variables below the threshold (determined by the expected sparsity) are mapped to 0, resulting in the final pruned structure.

Following previous work (Xia et al., 2022), we use Lagrangian terms that force the expected sparsity of the model to be close to the desired sparsity:

$$L_{\text{prune}} = \lambda_1 \cdot (\hat{s} - t) + \lambda_2 \cdot (\hat{s} - t)^2, \tag{13}$$

where $\hat{s}$ is the expected sparsity, t is the target sparsity, and λ_1, λ_2 are two Lagrange multipliers. Please refer to Appendix B for more detail of $\hat{s}$.

Distillation Objective. We also improve the performance of the student model through knowledge distillation. We use both output distillation and layer distillation. For the former one, we use cross-entropy to compare the outputs of the teacher and student models:

$$\mathcal{L}_{\text{pred}} = D_{\text{KL}}(\mathrm{p}_s \| \mathrm{p}_t). \tag{14}$$

For the layer distillation, we dynamically map layers between student and teacher models, comparing their outputs using MSE loss. Given $\mathcal{T}$ as the student model's layer set and $m(\cdot)$ as its i-layer corresponding to the teacher's $m(i)$-th layer, the distillation loss on layer output are defined as:

$$\begin{aligned}\mathcal{L}_{\text{layer}} &= \sum_{i\in\mathcal{T}} \mathrm{MSE}(W_{\text{layer}} \mathbf{H}_s^i, \mathbf{H}_t^{m(i)}), \\ m(i) &= \arg\min_{j:j\geq i} \mathrm{MSE}(W_{\text{layer}} \mathbf{H}_s^i, \mathbf{H}_t^j),\end{aligned} \tag{15}$$

where $W_{\text{layer}} \in R^{d\times d}$ is a linear transformation maxtrix, initialized as an indentity matrix. $\mathbf{H}_s^i$ are hidden states from the i-th FFN block of the student model, and $\mathbf{H}_t^{m(i)}$ are hidden states from the $m(i)$-th FFN block of the teacher model. The final distillation loss combines the two types of losses:

$$\mathcal{L}_{\text{distill}} = \lambda \mathcal{L}_{\text{pred}} + (1-\lambda)\mathcal{L}_{\text{layer}}, \tag{16}$$

where λ controls the contribution of each loss.

Multi-stage Training. In DA-WID , we use 3 levels of masks: dimension-level mask, head-level mask, and layer-level mask. We find that among the multiple-level masks of the model, the head-level mask and the layer-level mask are prioritized for compressing the model during the training process, while the dimension-level mask may be ignored. In order to fully utilize the mask at each level, we divide the training process into several stages. We first train the model with the distillation objective. Then, we add the pruning objective and use a scheduling program to linearly increase the targert sparsity to the final target sparsity. After adding the pruning objective, we use only dimension-level masks for the first few epochs, after which we add head-level masks and layer-level masks.

3.4 Compactor Matrices Fusing

After training, we first prune compactor matrices, attention heads, the MHA block, and the FFN block based on masks. After that, we merge the compactor matrices with weight matrices. In addition to this, since we adaptively learn different hidden dimensions in different layers, we need to add additional weight matrices to the residual. For example, for a MHA block, we have

$$x_o = \mathrm{LN}(W_{\mathrm{R}} x + \mathrm{MHA}(x))), \quad (17)$$

where W_{R} is a matrix added to the residual, which is determined by multiplying the pruned compactor matrices before and after the MHA block (refer to Figure 3), i.e.,

$$W_{\mathrm{R}}^{(i)} = \mathrm{Prune}(C_{\mathrm{in}}^{(\mathrm{next})} C_{\mathrm{out}}^{(\mathrm{prev})}, M_{\mathrm{in}}^{(\mathrm{next})}, M_{\mathrm{out}}^{(\mathrm{prev})}), \quad (18)$$

where $C_{\mathrm{out}}^{(\mathrm{prev})}$ and $M_{\mathrm{out}}^{(\mathrm{prev})}$ represent the compactor matrix and mask before the MHA block, and $C_{\mathrm{in}}^{(\mathrm{next})}$, $M_{\mathrm{in}}^{(\mathrm{next})}$ represent those after the MHA block; function Prune denotes pruning rows and columns of matrix $C_{\mathrm{in}}^{(\mathrm{next})} C_{\mathrm{out}}^{(\mathrm{prev})}$ via masks $M_{\mathrm{in}}^{(\mathrm{next})}$, $M_{\mathrm{out}}^{(\mathrm{prev})}$, respectively. The FFN block follows the same way.

4 Experiments

4.1 Setup

Datasets. We evaluate our approach on GLUE (Wang et al., 2018) tasks and SQuAD (Rajpurkar et al., 2016) v1.1. GLUE tasks include SST-2 (Socher et al., 2013), MNLI (Kim et al., 2019), QQP (Wang et al., 2017), QNLI, MRPC (Dolan and Brockett, 2005), STS-B, and RTE. We exclude CoLA due to their unstable behaviors, and we cannot reproduce some baseline results based on our device on the CoLA dataset. For the GLUE benchmark, we report accuracy for the MNLI, QQP, QNLI, SST2, MRPC, and RTE tasks, as well as spearman correlation for the STS-B task. For the SQuAD benchmark, we report the F1 score. For more comprehensive information regarding the experimental setup, please refer to Appendix C.

Baselines. We compare DA-WID with powerful distillation methods including TinyBERT (Jiao et al., 2020), WID (Wu et al., 2023) and the pruning method CoFi (Xia et al., 2022). For TinyBERT, we use the experimental results without data augmentation for a fair comparison.

4.2 Main Results

As shown in Table 1, we compress the $\mathrm{BERT}_{\mathrm{base}}$ model and compare the performance of DA-WID with other methods under the similar sparsity. First, compared to the original model, DA-WID retains 96% of the model performance while removing 94% of the model parameters. Second, compared to other baselines, DA-WID has extra degrees of freedom. Compared to the pruning method, DA-WID can additionally compress the hidden dimensions and intermediate dimensions of the MHA block. Compared to distillation methods, DA-WID can use different hidden dimensions at different layers with additional head-level and layer-level pruning. Overall, DA-WID obtains the best performance on all datasets, indicating that utilizing these extra degrees of freedom can benefit the performance during compression.

4.3 Ablation Study

Compactor Initialization. To verify the importance of the data-aware initialization, we initialize the compactor matrices to identity matrices and re-run the experiment. To prevent head-level and layer-level masks from interfering with the role of data-aware initialization, we use only dimension-level masks in our experiments. As shown in Table 2, on the RTE and MRPC datasets, removing data-aware initialization leads to significant performance degradation, while removing data-aware compactor initialization on the SST-2 and STS-B datasets also leads to slight performance degradation. This suggests that data-aware initialization of the compression matrix helps to improve the performance of the compressed model.

Data-aware initialization uses calibration data from the training set. We further explored the effect of the number of tokens on the performance of the model. Table 3 shows model accuracy with varying sample tokens. Since increasing the number of samples increases the complexity of computing the SVD, sampling 4,096 tokens offer a good balance between accuracy and computation.

Impact of Multi-level Masks. DA-WID uses different levels of masks to compress the model. In order to explore the impact of different levels of masks on model performance, we conduct the following experiments: (1) Use all levels of masks. (2) Ignore head-level masks. (3) Ignore layer-level masks. (4) Ignore head-level and layer-level masks.

	Params.	SST-2	QNLI	MNLI	QQP	RTE	STS-B	MRPC	SQuAD	Avg.
$\text{BERT}_{\text{base}}$	85M	93.1	91.5	84.8	91.2	70.4	89.1	85.6	88.4	86.76
TinyBERT_4	4.7M	89.7	86.7	78.8	90.0	63.2	85.0	81.4	82.1	82.11
WID	5.0M	88.8	85.4	78.4	89.5	60.3	84.5	81.9	81.2	81.25
CoFi	~5.0M	90.6	86.1	80.6	**90.1**	64.7	83.1	82.6	82.6	82.55
Ours	~5.0M	**91.4**	**87.6**	**81.6**	**90.1**	**66.4**	**86.1**	**82.8**	**83.2**	**83.65**

Table 1: Comparison between our DA-WID and both the distillation methods and pruning methods. Note that, following previous work (Xia et al., 2022), we do not count the number of parameters in the embedding layer.

	SST-2	RTE	STS-B	MRPC
DA-WID-10M	**91.3**	**66.8**	**87.8**	**85.5**
w/o initialize	91.2	52.7	85.0	80.6
DA-WID-5M	**91.4**	**66.4**	86.1	82.8
w/o head	90.9	66.1	**86.6**	82.1
w/o layer	91.2	62.8	86.1	83.5
w/o head & layer	91.1	63.9	86.5	**83.8**

Table 2: Ablation studies on compactor initialization and pruning units on SST-2, RTE, STS-B, and MRPC datasets. For DA-WID-10M, we ignored corse-grained masks $M_{\text{head}}, M_{\text{MHA}}$ and M_{FFN} and compressed the model to 10M. For DA-WID-5M, we used masks of all granularities and compressed the model to 5M.

Number of tokens	SST-2	RTE	STS-B	MRPC
2,048	90.0	58.8	85.9	82.8
4,096	**91.4**	**64.2**	86.1	82.8
8,192	90.8	63.5	**86.4**	**83.1**

Table 3: Ablation studies on the number of tokens sampled on SST-2, RTE, STS-B, and MRPC datasets.

The findings from the experiment are presented in Table 2. Observations indicate superior model performance on the SST-2 and RTE datasets when all mask levels are utilized. For the STS-B datasets, the removal of the head-level mask results in the most precise models. Notably, the optimal performance on the MRPC dataset is achieved by a model that excludes both the head-level and layer-level masks. Given the data volume in each dataset, we hypothesize that minor alterations in the head-level and layer-level masks can significantly influence model outputs compared to the dimension-level mask. This implies that the head-level and layer-level masks might be more challenging to optimize. Consequently, removing either the head-level or layer-level mask in smaller datasets can stabilize the optimization process, leading to a more precise model. As dataset sizes increase, the need for model compression flexibility becomes evident, with multi-level masking yielding superior outcomes.

4.4 Structures of Pruned Model

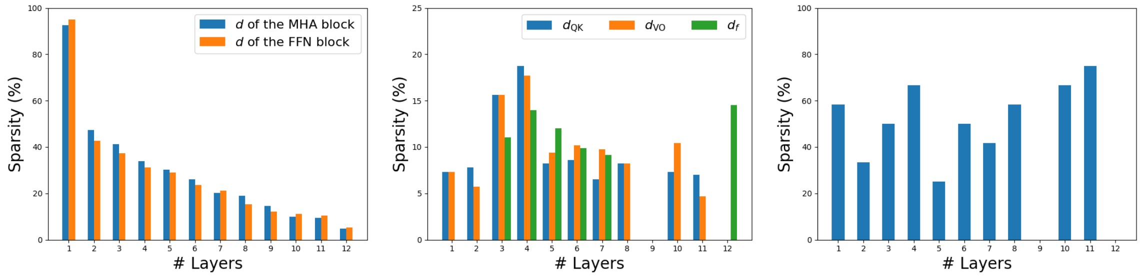
We study the pruned structures produced by DA-WID. Take the MRPC dataset as an example. Figure 4 shows the structural information of the pruned model, and the results of other datasets are shown in Appendix D.

From Fig. 4 (b) and (c), as well as related figures for other datasets, it's evident that the model structure varies across different datasets. However, a consistent observation across these structures is that layers nearer the output are more compressed than those closer to the inputs. Additionally, the intermediate dimensions of the FFN block are notably more compressed across all datasets compared to the intermediate dimensions of the MHA block. This distinction is highlighted when comparing the green bars to the blue and red bars in Fig. 4 (b).

The observed compression patterns align with models derived from previous pruning efforts as cited in (Xia et al., 2022). Besides these findings, which concur with the pruning method, Fig. 4 (a) and its analogous figures for other datasets reveal that the model's hidden dimension decreases as the number of layers increases. This suggests that the model progressively compresses features to more compact dimensions throughout its inference process.

5 Related Work

Distillation. Knowledge distillation (Hinton et al., 2015) is a model compression approach that transfers knowledge from a larger teacher model to a smaller student model. Most distillation methods assume a fixed student structure, and at the same time, pre-training of the student model from scratch on unlabeled corpora is important for these dis-



(a) Hidden dimensions

(b) Intermediate dimensions

(c) Attention head size

Figure 4: Structural information of the pruned model on the MRPC dataset, where sparsity denotes the ratio of the remaining dimension or size to the original dimension or size. (a) Output dimensions of each MHA and FFN block. (b) Intermediate dimensions of each MHA and FFN block. (c) The number of attention heads in each MHA block.

tillation methods, but this results in high computational costs. In addition to the above methods, DynaBERT (Hou et al., 2020) tries to distill the student model with adaptive width and height, and WID (Wu et al., 2023) inherits the parameters of the teacher model and tries to directly compress the teacher model into the student model through the re-parameterization method. Other methods, such as DistillBERT (Sanh et al., 2019), initialize the student model through the teacher model to avoid the pre-training phase, but these methods limit the possible model structures of the student model. In contrast to the above methods, our approach eliminates the need for a pre-training phase while allowing for adaptive determination of the student model structure.

Pruning. Existing pruning methods can be broadly divided into two categories: unstructured pruning and structured pruning. Unstructured pruning (Gale et al., 2019; Frankle and Carbin, 2018; Kurtic et al., 2022; Louizos et al., 2018; Sanh et al., 2020) aims to remove unimportant scalar values from the model's parameters. Although unstructured pruning algorithms can remove many redundant parameters while ensuring accuracy, compressed models require specific sparse data structures and hardware support to take advantage of unstructured pruning. For this reason, structure pruning approaches (Kwon et al., 2022; Lin et al., 2020; Lagunas et al., 2021; Sajjad et al., 2023; Wang et al., 2020; Xia et al., 2022) are proposed to remove weight blocks in PLM, including the entire layer (Fan et al., 2019; Prasanna et al., 2020; Sajjad et al., 2020), attention heads of the MHA block (Michel et al., 2019; Voita et al., 2019), and filters of the FFN block (McCarley et al., 2019). Structure pruning can accelerate inference speed and reduce memory overhead without specialized data structures and hardware. We introduce structured pruning to our approach to increase the model structure's flexibility.

Low-Rank Factorization. Some low-rank factorization work compresses PLM directly by decomposing the weight matrix (Liu and Ng, 2022; Yin et al., 2022; Zhou et al., 2019; Hua et al., 2022). Other works (Ma et al., 2019; Xiao et al., 2023) have considered the model structure of PLM while performing matrix decomposition, and these works are mainly used for the compression of MHA blocks. In our approach, instead of compressing PLMs directly using low-rank factorization, we initialize the parameters of the model by low-rank factorization. Besides initializing the compactor matrices in the MHA block through low-rank factorization, we also contemplate initializing compactor matrices to reduce the hidden dimensions between layers.

6 Conclusion

This study introduces DA-WID , an enhanced WID approach tailored for compressing PLMs. DA-WID employs a data-aware initialization, facilitating easier optimization of the compression model, thereby boosting its performance. Concurrently, DA-WID refines the WID-based structure and integrates it with a pruning technique. This allows the model to selectively determine its architecture in line with the desired sparsity. When applied to $\mathrm{BERT}_{\mathrm{base}}$ and evaluated on the GLUE and SQuAD benchmarks, DA-WID notably achieves a 94% sparsity with only a minor 4% reduction in accuracy.

7 Limitations

Our proposed DA-WID introduces extra weight matrices in the residual parts when merging the inserted compactor matrices with the weight matrices. When the model is compressed to 5M, these extra parameters account for more than half of the model. This predominance hinders further compression. In future research, we aim to explore strategies to eliminate these extraneous parameters.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization.

Patrick Chen, Hsiang-Fu Yu, Inderjit Dhillon, and Cho-Jui Hsieh. 2021. Drone: Data-aware low-rank compression for large nlp models. *Advances in neural information processing systems*, 34:29321–29334.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Angela Fan, Edouard Grave, and Armand Joulin. 2019. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*.

Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Training pruned neural networks. *CoRR*, abs/1803.03635.

Trevor Gale, Erich Elsen, and Sara Hooker. 2019. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33:9782–9793.

Ting Hua, Yen-Chang Hsu, Felicity Wang, Qian Lou, Yilin Shen, and Hongxia Jin. 2022. Numerical optimizations for weighted low-rank estimation on language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1404–1416, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling bert for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174.

Seonhoon Kim, Inho Kang, and Nojun Kwak. 2019. Semantic sentence matching with densely-connected recurrent and co-attentive information. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6586–6593.

Eldar Kurtic, Daniel Campos, Tuan Nguyen, Elias Frantar, Mark Kurtz, Benjamin Fineran, Michael Goin, and Dan Alistarh. 2022. The optimal BERT surgeon: Scalable and accurate second-order pruning for large language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4163–4181, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Woosuk Kwon, Sehoon Kim, Michael W Mahoney, Joseph Hassoun, Kurt Keutzer, and Amir Gholami. 2022. A fast post-training pruning framework for transformers. *arXiv preprint arXiv:2204.09656*.

François Lagunas, Ella Charlaix, Victor Sanh, and Alexander Rush. 2021. Block pruning for faster transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10619–10629, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zi Lin, Jeremiah Liu, Zi Yang, Nan Hua, and Dan Roth. 2020. Pruning redundant mappings in transformer models via spectral-normalized identity prior. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 719–730, Online. Association for Computational Linguistics.

Ye Liu and Michael K Ng. 2022. Deep neural network compression by tucker decomposition with nonlinear response. *Knowledge-Based Systems*, page 108171.

Christos Louizos, Max Welling, and Diederik P Kingma. 2018. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*.

Xindian Ma, Peng Zhang, Shuai Zhang, Nan Duan, Yuexian Hou, Ming Zhou, and Dawei Song. 2019. A tensorized transformer for language modeling. *Advances in neural information processing systems*, 32.

JS McCarley, Rishav Chakravarti, and Avirup Sil. 2019. Structured pruning of a bert-based question answering model. *arXiv preprint arXiv:1910.06360*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. *CoRR*, abs/1912.01703.

Sai Prasanna, Anna Rogers, and Anna Rumshisky. 2020. When bert plays the lottery, all tickets are winning. *arXiv preprint arXiv:2005.00561*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2020. Poor man's bert: Smaller and faster transformer models. *arXiv preprint arXiv:2004.03844*, 2(2).

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2023. On the effect of dropping layers of pre-trained transformer models. *Computer Speech & Language*, 77:101429.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Victor Sanh, Thomas Wolf, and Alexander Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in Neural Information Processing Systems*, 33:20378–20389.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2020. Structured pruning of large language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6151–6162.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Taiqiang Wu, Cheng Hou, Zhe Zhao, Shanshan Lao, Jiayi Li, Ngai Wong, and Yujiu Yang. 2023. Weight-inherited distillation for task-agnostic bert compression. *arXiv preprint arXiv:2305.09098*.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. 2022. Structured pruning learns compact and accurate models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1513–1528.

Jinqi Xiao, Miao Yin, Yu Gong, Xiao Zang, Jian Ren, and Bo Yuan. 2023. COMCAT: Towards efficient compression and customization of attention-based vision models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 38125–38136. PMLR.

Miao Yin, Huy Phan, Xiao Zang, Siyu Liao, and Bo Yuan. 2022. Batude: Budget-aware neural network compression based on tucker decomposition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 8, pages 8874–8882.

Mingyi Zhou, Yipeng Liu, Zhen Long, Longxi Chen, and Ce Zhu. 2019. Tensor rank learning in cp decomposition via convolutional neural network. *Signal Processing: Image Communication*, 73:12–21.

A Data-aware Compactor Initialization

A.1 Compactor matrices $W_Q^{(i)}, W_K^{(i)}$ Initialization

We initialize compactor matrices $C_Q^{(i)}$ and $C_K^{(i)}$ in Eq. 6 based on the solution of Eq. 8. The solving process for Eq. 8 can be found in DRONE (Chen et al., 2021). Assume that

$$U_Q^{(i)}, \Sigma_Q^{(i)}, V_Q^{(i)\top} = \mathrm{SVD}(W_Q^{(i)} X),$$
$$U_K^{(i)}, \Sigma_K^{(i)}, V_K^{(i)\top} = \mathrm{SVD}(W_K^{(i)} X),$$
$$M^{(i)} = \Sigma_Q^{(i)\top} U_Q^{(i)\top} U_K^{(i)} \Sigma_K^{(i)},$$
$$U_M^{(i)}, \Sigma_M^{(i)}, V_M^{(i)\top} = \mathrm{SVD}(M^{(i)}), \tag{19}$$

then we define

$$U^{(i)} = \Sigma_M^{(i)\frac{1}{2}} U_M^{(i)\top} \Sigma_Q^{(i)-1} U_Q^{(i)\top},$$
$$V^{(i)} = \Sigma_M^{(i)\frac{1}{2}} V_M^{(i)} \Sigma_K^{(i)-1} U_K^{(i)\top}, \tag{20}$$

and let $C_Q^{(i)} = U^{(i)}, C_K^{(i)} = V^{(i)}$.

A.2 Compactor matrices V, O Initialization

We initialize compactor matrices $C_V^{(i)}$ and $C_O^{(i)}$ in Eq. 6 based on the solution of Eq. 9. Eq. 9, which can be solved in the same way as Eq. 8. However, we also find a sub-optimal but more easily implementable way of solving this equation. Assume that

$$U^{(i)}, \Sigma^{(i)}, V^{(i)\top} = \mathrm{SVD}(W_V^{(i)} X S), \tag{21}$$

then we let $C_V^{(i)} = U^{(i)\top}, C_O^{(i)} = U^{(i)}$.

B Sparity

The expected sparsity $\hat{s}$ is computed as follow

$$\hat{s} = \frac{1}{M}($$
$$\sum_i^L \sum_j^H \sum_k^d \sum_l^{d_h} M_{\mathrm{MHA}}^{(i)} \cdot M_{\mathrm{head}}^{(i,j)} \cdot M_{\mathrm{out,FFN,i-1}}^{(i,k)} \cdot M_Q^{(i,l)} +$$
$$\sum_i^L \sum_j^H \sum_k^d \sum_l^{d_h} M_{\mathrm{MHA}}^{(i)} \cdot M_{\mathrm{head}}^{(i,j)} \cdot M_{\mathrm{out,FFN,i-1}}^{(i,k)} \cdot M_K^{(i,l)} +$$
$$\sum_i^L \sum_j^H \sum_k^d \sum_l^{d_h} M_{\mathrm{MHA}}^{(i)} \cdot M_{\mathrm{head}}^{(i,j)} \cdot M_{\mathrm{out,FFN,i-1}}^{(i,k)} \cdot M_V^{(i,l)} +$$
$$\sum_i^L \sum_j^H \sum_k^d \sum_l^{d_h} M_{\mathrm{MHA}}^{(i)} \cdot M_{\mathrm{head}}^{(i,j)} \cdot M_{\mathrm{in,MHA,i}}^{(i,k)} \cdot M_O^{(i,l)} +$$
$$\sum_i^L \sum_k^d \sum_l^{d_f} M_{\mathrm{FFN}}^{(i)} \cdot M_{\mathrm{out,MHA,i}}^{(i,k)} \cdot M_f^{(i,l)} +$$
$$\sum_i^L \sum_k^d \sum_l^{d_f} M_{\mathrm{FFN}}^{(i)} \cdot M_{\mathrm{in,FFN,i}}^{(i,k)} \cdot M_f^{(i,l)} +$$
$$\sum_i^L \sum_k^d M_{\mathrm{out,FFN,i-1}}^{(i,k)} \cdot M_{\mathrm{in,MHA,i}}^{(i,k)} +$$
$$\sum_i^L \sum_k^d M_{\mathrm{out,MHA,i}}^{(i,k)} \cdot M_{\mathrm{in,FFN,i}}^{(i,k)}), \tag{22}$$

where M denotes denotes the total number of parameters of PLM.

C Experiment Details

C.1 Experiment Setup

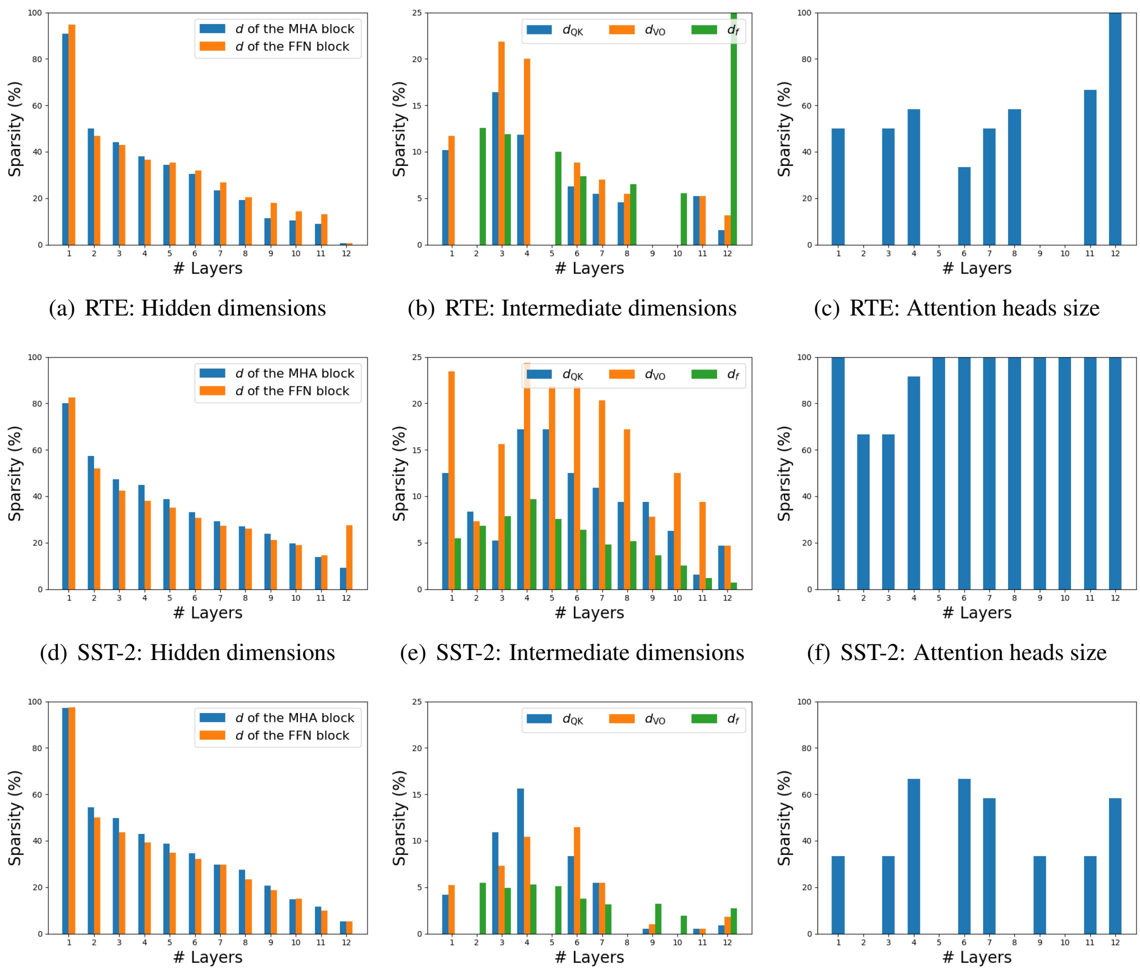
We implemented our method on top of PyTorch (Paszke et al., 2019) and used a single 3090 GPU for all experiments. To establish the baseline models, we first download the pre-trained checkpoints from the HuggingFace (Wolf et al., 2019) Transformers repository. For the BERT model, we conduct fine-tuning on the pre-trained model for 3 epochs, employing a batch size of 16, 24, and 32 and a learning rate of 1e-5 and 2e-5 for tasks in the GLUE benchmark and SQuAD dataset. Then, we sample 512 instances from the training data and sample 8 tokens for each instance to initialize compactor matrices. Finally, we fine-tune the model using the same settings utilized during the fine-tuning of the baseline models for 20 epochs. We start dimension-level pruning at the 2nd epoch; after that, we start head-level and layer-level pruning at the 8th epoch. Other parameters are set to the default parameters provided by the HuggingFace framework. To reduce memory usage we freeze the Embedding layer and the weight matrices in the MHA block and FFN block.

C.2 Datasets

GLUE (Wang et al., 2018) benchmark consists of various tasks related to sentence similarity calculation, sentence classification, textual entailment, and natural language inference. It includes 10 tasks, namely AX, COLA, QQP, MNLI, MRPC, QNLI, QQP, RTE, SST-2, STS-B, and WNLI. The number of training examples for each task is as follows: 1.1k, 10.7k, 432k, 5.8k, 105k, 364k, 3k, 70k, 67k, and 852, respectively. SQuAD 1.1 (Rajpurkar et al., 2016) dataset involves question and answer tasks, containing 88K training examples.

D Structures of Pruned Model

The structure of pruned models on RTE, SST-2, STS-B, MNLI, QNLI, QQP and SQuAD are shown in Fig. 5 and Fig. 6. We show the model structure in terms of dimension-level and head-level sparsity. Instead of directly showing the layer-level sparsity, we indirectly show the layer-level sparsity by dimension-level and head-level sparsity in the histograms, and if the value of a certain position is 0, it can be assumed that layer-level pruning has occurred at that position.

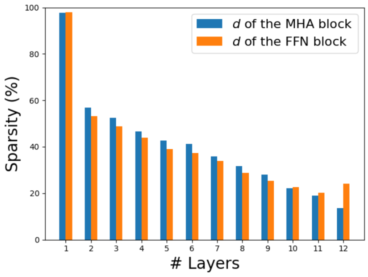


(a) RTE: Hidden dimensions (b) RTE: Intermediate dimensions (c) RTE: Attention heads size

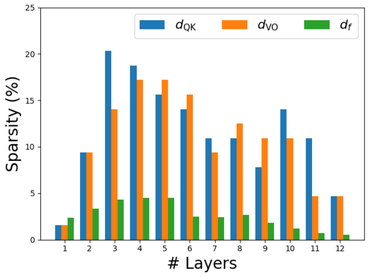
(d) SST-2: Hidden dimensions (e) SST-2: Intermediate dimensions (f) SST-2: Attention heads size

(g) STS-B: Hidden dimensions (h) STS-B: Intermediate dimensions (i) STS-B: Attention heads size

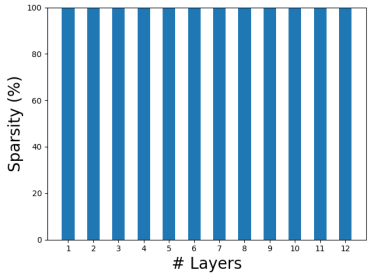
Figure 5: Pruned model structures on RTE, SST-2 and STS-B datasets



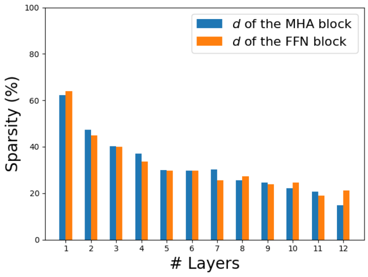
(a) MNLI: Hidden dimensions



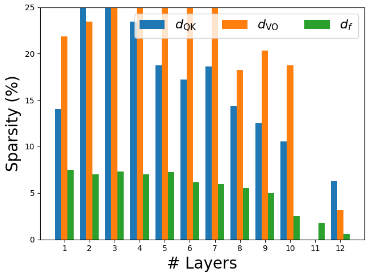
(b) MNLI: Intermediate dimensions



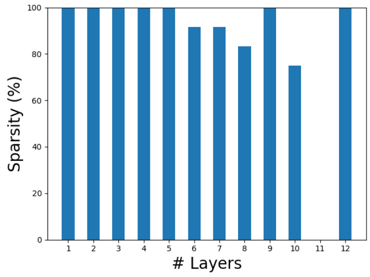
(c) MNLI: Attention heads size



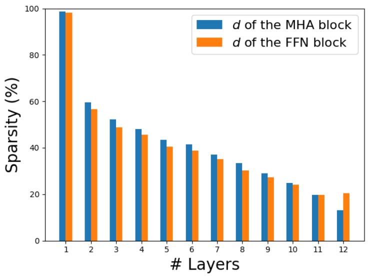
(d) QNLI: Hidden dimensions



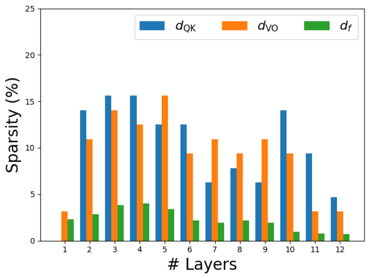
(e) QNLI: Intermediate dimensions



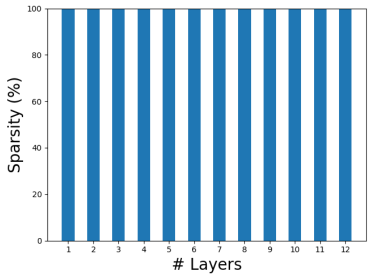
(f) QNLI: Attention heads size



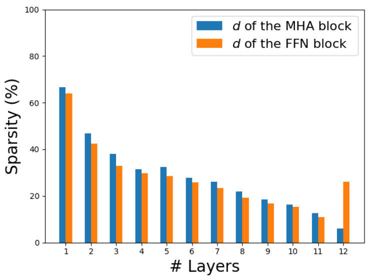
(g) QQP: Hidden dimensions



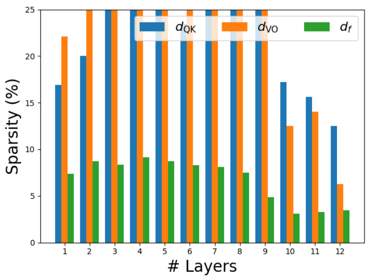
(h) QQP: Intermediate dimensions



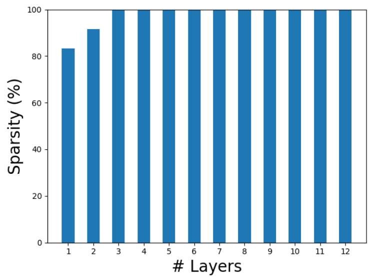
(i) QQP: Attention heads size



(j) SQuAD: Hidden dimensions



(k) SQuAD: Intermediate dimensions



(l) SQuAD: Attention heads size

Figure 6: Pruned model structures on MNLI, QNLI, QQP and SQuAD datasets